Appendix P

Rate Monotonic Analysis: *Did You Fake It?*

Content

Reprinted from the Software Program Manager's Network newsletter, NetFocus: Technology Update, *Number 210, June 1994*

P.1 Rate Monotonic Analysis

"In the past, developers have had few tools to help them ensure on-time performance of their real-time systems…A real-time system generally has several activities (or tasks), each of which must be completed by a specified deadline. Some of these deadlines may be hard (or critical) and some may be soft (such as those based on average performance). Missing a hard deadline can result in catastrophic loss of system performance or even loss of life." [OBENZA94]

P.2 RMA: Did You Fake It?

Did you fake it? This is the question concerned program and software managers should ask their development teams when it comes to the design and implementation of their real-time systems. Not only has "*faking it*" been an option when designing real-time systems, it has become the process. Unfortunately, there is immense corporate ignorance for the need of a proven analytical process to address real-time requirements at the design stage of a real-time system. With the exponential growth of the size of real-time systems, the decrease of development dollars, and the liability issues that are emerging due to defective software, faking it is no longer a viable option.

Schedulable, "*on time*" software is essential for real-time systems. Without such a capability, task timing conflicts cannot be identified early enough in development to avoid timing overruns, which cause system crashes and the subsequent need for budget-busting corrective measures. Simulation alone isn't the answer, for it doesn't guarantee timing, besides being extremely costly in time and money.

An alternative to faking it is Rate Monotonic Analysis (RMA). RMA is a scientific and mathematically sound way of guaranteeing the timing requirements of time-sensitive systems and is one of the most well known and often used scheduling algorithms for realtime applications. Boeing, General Electric, General Dynamics, Honeywell, IBM, McDonnell Douglas, Magnavox, Mitre, NASA, Naval Air Warfare Center, and Paramax are just some of the growing number of organizations beginning to use RMA on actual systems. RMA will increase your project's productivity and reduce integration and testing costs, risk and complexity.

RMA's system scheduling heuristic is shortest-task first, so the ready task with the shortest period always runs. The analysis considers worst-case time and combinations of system load, phasing, and resource consumption, ensuring real-time performance and stability under heavily loaded conditions. RMA provides insight into the hardware and software design that effects system timing performance and helps to identify possible bottlenecks and errors that degrade schedulability. RMA has the ability to predict timeliness — processing of an event during worst-case time — and guarantees that events meet their deadlines.

When you begin to integrate RMA into your process, you will become aware that the scheduling of tasks for a real-time system is significantly different from the traditional forms of scheduling. It will become apparent that tasks compete for resources; whether it's a CPU, backplane bus, or network. This requires that more important tasks be given priority to execute over all other tasks. A preemption occurs when a higher priority task replaces a currently running task on a resource. Preemption by way of priorities, to increase the responsiveness of a system, brings up several questions:

- How do you ensure beyond a reasonable doubt that the lower-priority tasks will ever get access to the resource?
- If there are simultaneous activities, how do you guarantee that all of them are completed on time?
- If they all cannot finish in a timely manner, which one succeed and which ones fail?
- Is it possible for all the tasks to run in a way that guarantees timely access to the resource?
- And most importantly, will the tasks meet their execution deadlines?

All of these questions can be answered through the use of RMA.

Adoption of RMA has been met with reluctance in the past. Questions arise about the costs of training, materials, and access to the appropriate tools that support RMA. Today, however, affordable CASE tools that support RMA are available, and provide a proven process as part of good engineering discipline in real-time systems. Unfortunately, management cannot promote good discipline if it doesn't thoroughly understand real-time system development issues. System engineering practitioners need the ability to analyze the run-time performance of a real-time system at all phases of the software life cycle:

- During the proposal phase there is a demand for support in system prototyping, iterative development techniques, and trade studies.
- Guaranteed timing deadlines and adequate system hardware are a necessity during system requirements definition — before writing a single line-of-code.
- At the time of implementation, the intent is to painlessly detect scheduling errors, calculate overall CPU usage, and diagnose corrective actions.
- The objective during test is to reduce testing, verification, and demonstration validation with the confidence that the algorithmic proof used at the design phase guarantees the system performance.

Achieving the objectives of real-time systems is possible if a system is designed with the support of a RMA CASE tool. CASE tools provide a practical, highly cost-effective, and easy way to automate use of RMA. Emerging RMA CASE tools take the guesswork out of identifying significant performance issues and can help develop an overall solution strategy as opposed to spending dollars on new hardware or using a less effective approach to modify a real-time system.

Adopting RMA methods as part of an organization's general engineering discipline and standard software development process will provide real cost savings and quality process improvements.

P.2.1 The "Did You Fake It?" Quiz

1. When it comes to real-time software, do you practice the method of first making the software work, and then trying to make it meet timing requirements?
2. Do you have unrepeatable system failures?
3. Have you ever bought a larger CPU to solve timing overruns?
4. Has rewriting your software in a different language been proposed as the answer to a timing error?
5. Do you wish you had a process on your current real-time program?
6. Do you think real-time means real fast?
7. Do you wish you had an accurate account of your real-time system's processing limitations?
8. Are you tempted to push more processing down to an interrupt level to get your system to run faster?

An affirmative answer to one or more of these questions indicates that your real-time systems are at high risk. Just say no to faking it by saying yes to the benefits of effectively using RMA.

P.3 RMA in Practice

A contractor had been directed by their customer to learn more about RMA and perform real-time analysis on the system they were building. They indicated that they did not need to perform any analysis because their simulation proved the system could meet timing requirements. By gathering information to understand the simulation, an RMA expert was able to extract the system design, analyze it, and identify a queue that was approaching overflow and would eventually bring the system down. The RMA expert asked the contractor to run the simulation an additional 10 minutes, which did cause the system to fail.

A Navy contractor had been building a submarine sonar trainer. At integration testing, the system was experiencing severe timing overruns, which were causing the system to crash. RMA showed only 300 lines-of-code had to be modified to fix the timing problems. This was considerably cheaper than recoding 17,000 lines of Ada to C, which was the original plan. The real-time analysis also showed that recoding to C would not have solved the problem.

Through the use of rate monotonic scheduling, we now have a system that will allow [Space Station] Freedom's computer's to budget their time, to choose [among] a variety of tasks, and decide not only which one to do first, but how much time to spend in the process. — Aaron Cohen, then Acting Deputy Administrator of NASA in 1992

RMA is derived from a paper presented by C.L. Liu and James Layland in 1973. In it they state that a set of *n* independent tasks will always meet its deadlines, for all task phasings, if:

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \le U(n) = n(2^{1/n} - 1)$$

C_1 = worst-case task execution time of task
T_1 = period of task
$U_{(n)}$ = utilization bound for *n* tasks

NOTE: For more information on RMA, see SEI's A Practitioners' Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems.

P.4 References

[OBENZA94] Obenza, Ray, "Guaranteeing Real-Time Performance Using RMA," *Embedded Systems Programming*, May 1994